What is claimed is:

1. A method of protecting memory locations associated with an embedded system, the method comprising:

starting a write filter that intercepts writes to the protected memory locations and stores the writes in a cache;

starting a state machine with at least a change state and a normal state;

upon starting the state machine, entering the change state when an indication is present that data needs to be persisted to the protected memory otherwise entering the normal state;

in the normal state identifying requests for critical writes to the protected memory and creating at least one update file describing the critical writes; and

in the change state, applying the changes in the update file and rebooting the system in a manner that persists the changes to the protected memory.

2. A method, as set forth in claim 1, further comprising:

emptying the cache upon startup of the embedded system.

3. A method, as set forth in claim 1, further comprising:

running applications of the embedded system in the normal state; and

not running the applications of the embedded system in the change state.

4. A method, as set forth in claim 1, wherein the step of applying the changes in the update file and rebooting comprises:

determining whether the application of the changes was successful;

if the application of the changes was not successful, deleting the update files, erasing the indication, issuing a command to empty the cache, and rebooting the embedded system; and

if the application of the changes was successful, issuing a command to persist the cache, deleting the update files, erasing the indication, and rebooting the system.

5. A method as set forth in claim 1, further comprising:

in the normal state when an update file is created, creating an indication that data needs to be persisted to the protected memory is set.

6. A method, as set forth in claim 5, wherein the step of creating an indication comprises writing the file name of the update file to a data file.

7. A method, as set forth in claim 1, wherein the step of creating an update file further comprises naming the update file using a time stamp.

8. A method, as set forth in claim 1, further comprising:

in the change state, if an update executable exists running said update executable.

9. A method, as set forth in claim 8, further comprising:

putting the state machine in a sleep mode during the execution of the update executable.

10. An embedded system comprising:

a processing unit responsive to an operating system for executing applications to perform the functions of the embedded system;

a main memory location storing the operating system of the embedded system, said operating system providing a write filter that protects the operating system from writes;

a secondary memory location for storing software and data; and

a control program that executes automatically upon booting of the system, said control program causing the embedded system to operate in a normal state and a change state, wherein:

during operation in the normal state, the application are run and when a critical write is requested, an update file is generated to store the critical write until the embedded system enters the change state; and

during operation in the change state, no applications are run and the update file is used to update and persist the operating system.

11. As embedded system that in conjunction with booting assume one of two states: a normal state in which application are executed and a write filter intercepts writes to a protected memory location and redirects them to a non-protected memory location; and a change state, entered across a boot from the normal state, in which writes applied to the write filter during the last normal state are re-applied to the write filter and subsequently persisted to the protected memory.

12. The embedded system, as set forth in claim 11, wherein applications are not run during the change state.

13. The embedded system, as set forth in claim 11, wherein only critical writes are applied to the write filter and persisted in the change state.

14. The embedded system, as set forth in claim 13, wherein critical write include writes to a system registry.

15. The embedded system, as set forth in claim 11, wherein writes intercepted by the write filter in the normal state are copied to at least one update file and in the change state the at least one update file is used as the source for re-applying the write to the write filter.

16. The embedded system, as set forth in claim 11, wherein during the change state updates to an operating system of the embedded system are applied and persisted.

17. The embedded system, as set forth in claim 11, wherein the updates are not applied during the normal state.

18. The embedded system, as set forth in claim 11, wherein the change state is entered subsequent to a boot when indicators of updates are present.

19. The embedded system, as set forth in claim 11, wherein once all updates have been persisted, the state machine enters the normal state.

20. The embedded system, as set forth in claim 11, wherein the protected memory location stores an operating system of the embedded system.